PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-161169

(43)Date of publication of application : 21.06.1996

(51)Int.Cl. G06F 9/38
G06F 9/28
G06F 9/45

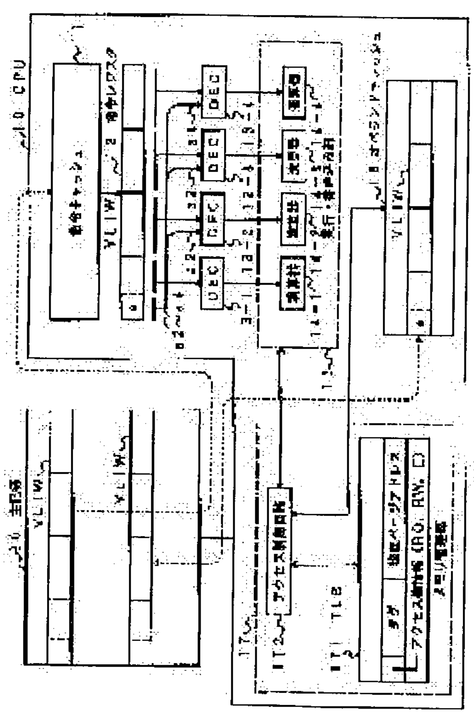
(21)Application number : 06-305907

(22)Date of filing : 09.12.1994

(71)Applicant : TOSHIBA CORP

(72)Inventor : TAKEUCHI YOICHIRO

(54) VLIW TYPE COMPUTER SYSTEM AND METHOD FOR INTERPRETING/ EXECUTING VLIW



(57)Abstract:

PURPOSE: To reduce a useless storage area due to a code part conventionally occupied by a nop instruction on an VLIW (very long instruction word).

CONSTITUTION: Istruction validating information (a) consisting of bits a2 to a4 indicating which instruction field (invalid instruction field) does not require interpretation/execution out of a 2nd instruction field and after is formed on a part of the leading field of a VLIW having four instruction fields, and when the VLIW is fetched in an instruction register 12, the contents (excluding the information (a)) of the 1st instruction field are unconditionally interpreted by a decoder 13-1 to control a computing element 14-1 and the contents of the 2nd to 4th instruction fields are interpreted by respective decoders 13-2 to 13-4 in accordance with the states of the bits a2 to a4 in the information (a) to control computing elements 14-2 to 14-4, so that the invalid instruction field part can be utilized as a variable/constant area.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

* NOTICES *

Japan Patent Office is not responsible for any damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] It is one long instruction word VLIW with the instruction field which the same number for controlling separately two or more computing elements in which juxtaposition actuation is possible, and each [these] computing element became independent of. The instruction register for holding the instruction word VLIW which has the instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in the part are, and then should be interpreted and performed, Corresponding to each instruction field of said instruction word VLIW currently held at this instruction register, it is prepared, respectively. It is a decoding means to interpret the contents of the corresponding instruction field and to control said computing element corresponding to the field concerned. When it is shown by said instruction validation information that the field concerned is an invalid instruction field A decoding means to forbid actuation of said corresponding computing element according to the contents of the field concerned, The computing system of the VLIW method characterized by providing a memory management means to allow accessing as an operation operand, to the instruction field on said instruction word VLIW the invalid thing is indicated to be using said instruction validation information.
[Claim 2] The computer system of the VLIW method according to claim 1 which is a compile means to generate the object program which consists of a train of the instruction word VLIW which compiles a source program and is performed within said computer system, and the interpretation and activation in said instruction word VLIW make an unnecessary invalid instruction field an operation operand field, and is characterized by providing further the compile means which assigns a constant or a variable to the field concerned.
[Claim 3] In the computing system of a VLIW method controlled by one long instruction word VLIW with the instruction field which the same number became independent of about two or more computing elements in which juxtaposition actuation is possible It is a link means to link the object group which consists of a train of said instruction word VLIW, and to generate a load module. A link means to delete the invalid instruction-field part except the effective instruction field which operates said computing element to the generate time of the load module concerned, When the demand page access request for loading the train of the instruction word VLIW of an assignment page on a primary storage from said load module occurs The computing system of the VLIW method characterized by providing a demand page access means to restore the train of the instruction word VLIW containing said invalid instruction-field part.
[Claim 4] Said instruction word VLIW has the instruction validation information that the invalid instruction field which does not have the need for an interpretation and activation in the part shows any they are. Said link means For said every instruction word VLIW, the location of said invalid instruction field is checked based on the instruction validation information on the VLIW, and the field concerned is removed. Said demand page access means In case the train of the instruction word VLIW of an assignment page is loaded on a primary storage from said load module, the instruction validation information on said instruction word VLIW is referred to. The computing system of the VLIW method

according to claim 3 characterized by detecting the location of said invalid instruction field on the VLIW concerned, flying the location, and arranging the contents of the effective instruction field on the VLIW concerned to said primary storage.
[Claim 5] Said link means generates the instruction map in which said arrangement of the effective instruction field and an invalid instruction field in the train of said instruction word VLIW before a link is shown, and adds it to said load module at said linking time. In case said demand page access means loads the train of the instruction word VLIW of an assignment page on a primary storage from said load module The computing system of the VLIW method according to claim 3 characterized by detecting the location of said invalid instruction field and arranging a no operation instruction nop in the location according to said instruction map given to the load module concerned.
[Claim 6] In the interpretation / activation approach of VLIW applied to the computing system of a VLIW method controlled by one long instruction word VLIW with the instruction field which the same number became independent of about two or more computing elements in which juxtaposition actuation is possible The instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in said a part of instruction word VLIW are is established. When the instruction field the invalid thing is indicated to be using said instruction validation information on the VLIW concerned on said instruction word VLIW which should be interpreted and performed exists As opposed to the instruction field on said VLIW with which forbid actuation of said corresponding computing element according to the contents of the instruction field, and the invalid thing is indicated to be using said instruction validation information The interpretation / activation approach of VLIW characterized by allowing the use as an operation operand field of the field concerned by allowing accessing as an operation operand.

[Translation done.]

* NOTICES *

TECHNICAL FIELD

[Industrial Application] This invention relates to the computing system of the VLIW method which controls two or more computing elements in which juxtaposition actuation is possible by one long instruction word called VLIW (Very Long Instruction Word) which is the independent assembly of two or more instructions (instruction field), and the interpretation / activation approach of VLIW.

[Translation done.]

PRIOR ART

[Description of the Prior Art] In recent years, the computing system of a VLIW method is developed for improvement in the speed of processing. each instruction with which this kind of computing system is equipped with two or more computing elements in which juxtaposition actuation is possible, and it became independent on VLIW -- each computing element concerned -- a synchronization -- taking -- each -- and it controls to juxtaposition.
[0003] Drawing 13 shows the basic configuration of the important section of the computing system of a VLIW method. In the system of this drawing 13 , it has composition with which four computing elements 132-1 to 132-4 are controlled by VLIW131 which makes four instructions of instruction #1-#4 1 set. namely, the system of drawing 13 -- setting -- the inside of VLIW131 -- each -- instruction #1-#4 -- respectively -- a decoder (DEC) 133-1 to 133-4 -- coincidence -- and it is interpreted by juxtaposition and actuation of the computing element 132-1 to 132-4 which corresponds, respectively is controlled.
[0004] Thus, in the computing system of a VLIW method, juxtaposition actuation of two or more computing elements can be carried out by VLIW. Therefore, a programmer or a compiler investigates the operation which can be performed to juxtaposition, by programming these activation to the instruction with which it corresponds on VLIW, parallel processing of instruction word level is realized and improvement in the speed of processing can be attained.
[0005] Now, in the program (object program) of the VLIW method used with this kind of computer system, the case which cannot use for an operation two or more computing elements of all in which juxtaposition actuation is originally possible depending on the causal relation (dependency during each instruction) of the computational algorithm to perform is generated.
[0006] In such a case, the no operation instruction (nop instruction) which specifies not calculating is arranged at the instruction field in VLIW corresponding to the computing element which is not used. Malfunction prevention is achieved by arrangement of this nop instruction.

[Translation done.]

EFFECT OF THE INVENTION

[Effect of the Invention] As explained in full detail above, the code part which was conventionally occupied with the nop instruction on VLIW according to this invention can be used now as a variable or a constant area, and the main storage areas which an activation object uses by this can be reduced.
[0118] Moreover, the auxiliary storage areas which a load module file occupies can be reduced by generating the load module with which a part for the invalid (unnecessary) instruction part conventionally occupied with the nop instruction on VLIW (instruction field) was removed according to this invention, and normal program manipulation can be performed by restoring to the original VLIW train at the time of the program load which moreover follows a demand page access request. According to such this invention, the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW can be reduced.

[Translation done.]

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] As described above, in the computing system of a VLIW method, a nop instruction is arranged for the instruction which can be executed to coincidence according to the dependency during each instruction a predetermined number ******** case for the malfunction prevention to some instruction fields in VLIW.
[0008] For this reason, when the number of computing elements is increased and the instruction word length of VLIW is extremely lengthened according to it, or when the processing which cannot carry out juxtaposition actuation easily from the first is programmed, most activation codes serve as a nop instruction, and it is in the end. There was a problem that an object program size (activation object size) became remarkably large, and the main storage area which the program occupies also became large. This was the same also about the executable load module on a disk (secondary memory).
[0009] In case code size is reduced and VLIW is interpreted and (decoding) performed by deleting the part of a nop instruction by making instruction format of VLIW into variable length although this problem is solved, the method restored to an original format in hardware can be considered. However, since a hardware configuration becomes very complicated and the engine performance also worsens, this method is not practical. And implementation of program control structure, such as count of a branching place, becomes very complicated.
[0010] This invention was made in consideration of the above-mentioned situation, and the purpose is in offering the computing system of the VLIW method which can reduce the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW, and the interpretation / activation approach of VLIW.
[0011] Other purposes of this invention are to enable it to reduce the main storage areas which can use the code part conventionally occupied with the nop instruction on VLIW as a variable or a constant area, have it, and an activation object uses. The purpose of further others of this invention is to be able to be made to make size of a load module file small.

[Translation done.]

OPERATION

[Means for Solving the Problem and its Function] Two or more computing elements which the configuration concerning the 1st viewpoint of this invention can juxtaposition operate, It is VLIW with the instruction field which the same number for controlling each [these] computing element separately became independent of. The instruction register for holding VLIW which has the instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in the part are, and then should be interpreted and performed, Corresponding to each instruction field of VLIW currently held at this instruction register, it is prepared, respectively. It is a decoding means to interpret the contents of the corresponding instruction field and to control the computing element corresponding to the field concerned. When it is shown by the above-mentioned instruction validation information that the field concerned is an invalid instruction field A decoding means to forbid actuation of the corresponding computing element according to the contents of the field concerned, It is characterized by having a memory management means to allow accessing as an operation operand, to the instruction field on VLIW the invalid thing is indicated to be using the above-mentioned instruction validation information. It is.
[0013] In the configuration concerning the 1st viewpoint of the above, when the fetch of the VLIW which should be performed next to an instruction register is carried out, about the contents of the field where it is shown using the instruction validation information that a part of the VLIW is made that it is the effective instruction field, a decoding means interprets (decoding) and controls actuation of the computing element which corresponds according to the interpretation result. Moreover, about the contents of the field where it is shown using instruction validation information that it is an invalid instruction field, it does not interpret and a corresponding computing element is not operated. In addition, although the interpretation itself carries out, the interpretation result is disregarded and you may make it not operate a corresponding computing element.
[0014] Thus, the computing element corresponding to the invalid instruction field in VLIW is forbidden from operating according to the contents of the invalid instruction field by control of the decoding means based on the instruction validation information on the VLIW concerned. For this reason, malfunction is not caused even if it uses that invalid instruction field as a field of an operation operand, without arranging a nop instruction to that invalid instruction field unlike the former. In this case, the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW can be reduced. In order to use the invalid instruction field in VLIW as a field of an operation operand here, in case the object program which compiles a source program and consists of a train of VLIW is generated, the interpretation and activation in VLIW should just assign a constant or a variable to an unnecessary invalid instruction field.
[0015] The configuration concerning the 2nd viewpoint of this invention is VLIW with the instruction field which the same number became independent of about two or more computing elements in which juxtaposition actuation is possible. It is a link means to link the object group which consists of a train of VLIW in the computer system of a VLIW method to control, and to generate a load module. A link means to delete the invalid instruction-field part except the effective instruction field which operates the

above-mentioned computing element to the generate time of the load module concerned, When the demand page access request for loading the train of VLIW of an assignment page on a primary storage from the above-mentioned load module occurs, it is characterized by having a demand page access means to restore the train of VLIW containing the above-mentioned invalid instruction-field part.
[0016] Unlike the former, in the configuration concerning the 2nd viewpoint of the above, the instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in the part are is prepared in each VLIW which constitutes the object group used as the candidate for a link by the link means.
[0017] A link means generates the load module with which the invalid instruction field was deleted by checking the location of an invalid instruction field based on the instruction validation information on the VLIW, and removing the field concerned for every VLIW. It enables this to reduce the size of the file on which a load module is put.
[0018] On the other hand, a demand page access means loads the train of VLIW of an assignment page on a primary storage from the above-mentioned load module, if a demand page access request occurs. However, the invalid instruction-field part is not contained in the load module concerned.
[0019] Then, in case a demand page access means loads the train of VLIW of an assignment page on a primary storage, with reference to the instruction validation information on corresponding VLIW, it detects the location of said invalid instruction field on the VLIW concerned, flies the location, and arranges the contents of the effective instruction field on the VLIW concerned to a primary storage. Thereby, VLIW is restored and arranged on a primary storage.
[0020] In addition, it is possible to realize reduction of the size of the load module by deletion of an invalid instruction field (nop instruction) and restoration of VLIW at the time of demand page access about VLIW of the same format as the former without instruction validation information, i.e., VLIW by which a nop instruction is set as an invalid instruction field.
[0021] What is necessary is for that just to generate first the load module with which the nop instruction was deleted for every VLIW at the linking time by the link means by detecting the nop instruction in that VLIW and removing that nop instruction. Moreover, for restoration of VLIW at the time of demand page access, the instruction map in which arrangement of the nop instruction (invalid instruction field set up) in the train of VLIW and the other instruction (effective instruction field set up) is shown is generated, and it adds to the load module at the linking time by the link means.
[0022] In this case, in case it loads the train of VLIW of an assignment page on a primary storage from a load module, according to the instruction map given to the load module concerned, a demand page access means detects the location of a nop instruction, is arranging a nop instruction in that location, and can restore and arrange the original VLIW on a primary storage.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

EXAMPLE

[Example] Hereafter, with reference to a drawing, it explains per example of this invention.

[Translation done.]

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] The block diagram showing the configuration of the computer system of the VLIW method concerning the 1st example of this invention.
[Drawing 2] Drawing showing an instruction format of VLIW applied in this example.
[Drawing 3] The flow chart for explaining the operand access processing by the memory management section 17 in drawing 1 .
[Drawing 4] Drawing showing the configuration of the compiler which generates the object program containing the VLIW group of the format shown in drawing 2 applied with the computer system of drawing 1 from a source program.
[Drawing 5] Drawing showing a part of flow chart for explaining the compile processing by the compiler of the configuration of drawing 4 .
[Drawing 6] Drawing showing the remainder of the flow chart for explaining the compile processing by the compiler of the configuration of drawing 4 .
[Drawing 7] Drawing showing the 2nd example aiming at reduction of the auxiliary storage areas which the file concerned occupies by the ability to be made to make small size of the load module file of a VLIW method.
[Drawing 8] The flow chart for explaining the link processing by the linkage editor 51 in drawing 7 .
[Drawing 9] Drawing showing a part of flow chart for explaining the demand page access processing by the virtual-memory-management section 52 in drawing 7 .
[Drawing 10] Drawing showing the remainder of the flow chart for explaining the demand page access processing by the virtual-memory-management section 52 in drawing 7 .
[Drawing 11] It is drawing in which it is drawing for explaining actuation of the configuration of drawing 7 , and drawing 11 (a) shows the situation of the invalid instruction-field deletion from VLIW at the time of link processing, and drawing 11 (b) shows the situation of the VLIW restoration at the time of demand page access processing.
[Drawing 12] Drawing showing the 3rd example aiming at reduction of the auxiliary storage areas which the file concerned occupies by the ability to be made to make small size of the load module file of a VLIW method.
[Drawing 13] Drawing showing the important section configuration of the computing system of the conventional VLIW method.
[Description of Notations]
10 [-- Decoder (DEC),] -- CPU, 11 -- An instruction cache, 12 -- An instruction register, 13-1 to 13-4 14-1 to 14-4 -- A computing element, 15 -- The activation / write-in section, 16 -- Operand cache, 17 -- The memory management section, 171 -- TLB, 172 -- Access-control circuit, 20 -- A primary storage, 31 -- A source program, 32 -- The 1st object program, 33 -- The 2nd object program, 41 -- The 1st compile section, 42 -- The 2nd compile section, 321,331 -- A code section, 322,332 -- Data division, 51 71 -- 52 A linkage editor (link means), 72 -- Virtual-memory-management section (demand page access means), 53 73 [-- A load module file, 621,821 / -- A code section, 622,822 / -- An index part, 821a / --

A compressed code field, 821b / -- Instruction map field.] -- 61 A primary storage, 81 -- 62 An object group, 82 -- 63 A load module, 83

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application] This invention relates to the computing system of the VLIW method which controls two or more computing elements in which juxtaposition actuation is possible by one long instruction word called VLIW (Very Long Instruction Word) which is the independent assembly of two or more instructions (instruction field), and the interpretation / activation approach of VLIW.
[0002]
[Description of the Prior Art] In recent years, the computing system of a VLIW method is developed for improvement in the speed of processing. each instruction with which this kind of computing system is equipped with two or more computing elements in which juxtaposition actuation is possible, and it became independent on VLIW -- each computing element concerned -- a synchronization -- taking -- each -- and it controls to juxtaposition.
[0003] Drawing 13 shows the basic configuration of the important section of the computing system of a VLIW method. In the system of this drawing 13 , it has composition with which four computing elements 132-1 to 132-4 are controlled by VLIW131 which makes four instructions of instruction #1-#4 1 set. namely, the system of drawing 13 -- setting -- the inside of VLIW131 -- each -- instruction #1-#4 -- respectively -- a decoder (DEC) 133-1 to 133-4 -- coincidence -- and it is interpreted by juxtaposition and actuation of the computing element 132-1 to 132-4 which corresponds, respectively is controlled.
[0004] Thus, in the computing system of a VLIW method, juxtaposition actuation of two or more computing elements can be carried out by VLIW. Therefore, a programmer or a compiler investigates the operation which can be performed to juxtaposition, by programming these activation to the instruction with which it corresponds on VLIW, parallel processing of instruction word level is realized and improvement in the speed of processing can be attained.
[0005] Now, in the program (object program) of the VLIW method used with this kind of computer system, the case which cannot use for an operation two or more computing elements of all in which juxtaposition actuation is originally possible depending on the causal relation (dependency during each instruction) of the computational algorithm to perform is generated.
[0006] In such a case, the no operation instruction (nop instruction) which specifies not calculating is arranged at the instruction field in VLIW corresponding to the computing element which is not used. Malfunction prevention is achieved by arrangement of this nop instruction.
[0007]
[Problem(s) to be Solved by the Invention] As described above, in the computing system of a VLIW method, a nop instruction is arranged for the instruction which can be executed to coincidence according to the dependency during each instruction a predetermined number ******** case for the malfunction prevention to some instruction fields in VLIW.
[0008] For this reason, when the number of computing elements is increased and the instruction word length of VLIW is extremely lengthened according to it, or when the processing which cannot carry out juxtaposition actuation easily from the first is programmed, most activation codes serve as a nop

instruction, and it is in the end. There was a problem that an object program size (activation object size) became remarkably large, and the main storage area which the program occupies also became large. This was the same also about the executable load module on a disk (secondary memory).

[0009] In case code size is reduced and VLIW is interpreted and (decoding) performed by deleting the part of a nop instruction by making instruction format of VLIW into variable length although this problem is solved, the method restored to an original format in hardware can be considered. However, since a hardware configuration becomes very complicated and the engine performance also worsens, this method is not practical. And implementation of program control structure, such as count of a branching place, becomes very complicated.

[0010] This invention was made in consideration of the above-mentioned situation, and the purpose is in offering the computing system of the VLIW method which can reduce the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW, and the interpretation / activation approach of VLIW.

[0011] Other purposes of this invention are to enable it to reduce the main storage areas which can use the code part conventionally occupied with the nop instruction on VLIW as a variable or a constant area, have it, and an activation object uses. The purpose of further others of this invention is to be able to be made to make size of a load module file small.

[0012]

[Means for Solving the Problem and its Function] Two or more computing elements which the configuration concerning the 1st viewpoint of this invention can juxtaposition operate, It is VLIW with the instruction field which the same number for controlling each [these] computing element separately became independent of. The instruction register for holding VLIW which has the instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in the part are, and then should be interpreted and performed, Corresponding to each instruction field of VLIW currently held at this instruction register, it is prepared, respectively. It is a decoding means to interpret the contents of the corresponding instruction field and to control the computing element corresponding to the field concerned. When it is shown by the above-mentioned instruction validation information that the field concerned is an invalid instruction field A decoding means to forbid actuation of the corresponding computing element according to the contents of the field concerned, It is characterized by having a memory management means to allow accessing as an operation operand, to the instruction field on VLIW the invalid thing is indicated to be using the above-mentioned instruction validation information.

[0013] In the configuration concerning the 1st viewpoint of the above, when the fetch of the VLIW which should be performed next to an instruction register is carried out, about the contents of the field where it is shown using the instruction validation information that a part of the VLIW is made that it is the effective instruction field, a decoding means interprets (decoding) and controls actuation of the computing element which corresponds according to the interpretation result. Moreover, about the contents of the field where it is shown using instruction validation information that it is an invalid instruction field, it does not interpret and a corresponding computing element is not operated. In addition, although the interpretation itself carries out, the interpretation result is disregarded and you may make it not operate a corresponding computing element.

[0014] Thus, the computing element corresponding to the invalid instruction field in VLIW is forbidden from operating according to the contents of the invalid instruction field by control of the decoding means based on the instruction validation information on the VLIW concerned. For this reason, malfunction is not caused even if it uses that invalid instruction field as a field of an operation operand, without arranging a nop instruction to that invalid instruction field unlike the former. In this case, the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW can be reduced. In order to use the invalid instruction field in VLIW as a field of an operation operand here, in case the object program which compiles a source program and consists of a train of VLIW is generated, the interpretation and activation in VLIW should just assign a constant or a variable to an unnecessary invalid instruction field.

[0015] In the computing system of a VLIW method controlled by VLIW in which the configuration concerning the 2nd viewpoint of this invention has the instruction field which the same number became independent of about two or more computing elements in which juxtaposition actuation is possible It is a link means to link the object group which consists of a train of VLIW, and to generate a load module. A link means to delete the invalid instruction-field part except the effective instruction field which operates the above-mentioned computing element to the generate time of the load module concerned, When the demand page access request for loading the train of VLIW of an assignment page on a primary storage from the above-mentioned load module occurs, it is characterized by having a demand page access means to restore the train of VLIW containing the above-mentioned invalid instruction-field part.
[0016] Unlike the former, in the configuration concerning the 2nd viewpoint of the above, the instruction validation information which shows any the invalid instruction fields which do not have the need for an interpretation and activation in the part are is prepared in each VLIW which constitutes the object group used as the candidate for a link by the link means.
[0017] A link means generates the load module with which the invalid instruction field was deleted by checking the location of an invalid instruction field based on the instruction validation information on the VLIW, and removing the field concerned for every VLIW. It enables this to reduce the size of the file on which a load module is put.
[0018] On the other hand, a demand page access means loads the train of VLIW of an assignment page on a primary storage from the above-mentioned load module, if a demand page access request occurs. However, the invalid instruction-field part is not contained in the load module concerned.
[0019] Then, in case a demand page access means loads the train of VLIW of an assignment page on a primary storage, with reference to the instruction validation information on corresponding VLIW, it detects the location of said invalid instruction field on the VLIW concerned, flies the location, and arranges the contents of the effective instruction field on the VLIW concerned to a primary storage. Thereby, VLIW is restored and arranged on a primary storage.
[0020] In addition, it is possible to realize reduction of the size of the load module by deletion of an invalid instruction field (nop instruction) and restoration of VLIW at the time of demand page access about VLIW of the same format as the former without instruction validation information, i.e., VLIW by which a nop instruction is set as an invalid instruction field.
[0021] What is necessary is for that just to generate first the load module with which the nop instruction was deleted for every VLIW at the linking time by the link means by detecting the nop instruction in that VLIW and removing that nop instruction. Moreover, for restoration of VLIW at the time of demand page access, the instruction map in which arrangement of the nop instruction (invalid instruction field set up) in the train of VLIW and the other instruction (effective instruction field set up) is shown is generated, and it adds to the load module at the linking time by the link means.
[0022] In this case, in case it loads the train of VLIW of an assignment page on a primary storage from a load module, according to the instruction map given to the load module concerned, a demand page access means detects the location of a nop instruction, is arranging a nop instruction in that location, and can restore and arrange the original VLIW on a primary storage.
[0023]
[Example] Hereafter, with reference to a drawing, it explains per example of this invention.
The block diagram showing the configuration of the computer system of the VLIW method which [1st example] drawing 1 requires for the 1st example of this invention, and drawing 2 are drawings showing an instruction format of VLIW applied in this example.
[0024] First, the instruction word length of VLIW applied by this example is 128 bits. This 128-bit VLIW is divided into four 32-bit instruction fields as shown in drawing 2 .
[0025] the high order triplet (it is the triplet of most left-hand side on a drawing) of the 1st instruction fields (it is the instruction field of most left-hand side on a drawing) of VLIW -- bit a2 -a4 from -- it is used as the setting field (instruction validation field) of the becoming instruction validation information a, and is used for the remaining 29 bits arranging the 1st instruction (instruction #1) of VLIW. Moreover, the 2nd - the 4th instruction field of VLIW is used for arranging the 4th instruction [the 2nd -

] (instruction #2-#4) of VLIW.
[0026] Bit a2 -a4 which constitutes the instruction validation information a (The case of "0") is shown for whether it is the effective instruction in VLIW with which the contents of a setting of the 2nd - the 4th instruction field have the need for an interpretation and activation, respectively (when it is "1"). In this example, when the bit ai (i=2-4) of the instruction validation information a is "0", the i-th instruction field in VLIW is available as an operation operand field (a variable/constant area is called hereafter) where a variable or a constant is set up.
[0027] In the computing system of drawing 1 , CPU to which 10 makes the center of a system, and 20 are primary storages in which the program (object program) of a VLIW method, an operand, etc. are stored.
[0028] CPU10 is equipped with the decoder (DEC) 13-1 to 13-4 which interprets the contents of a setting of the 1st - the 4th instruction field in VLIW held at the instruction cache 11 on which some counterparts of the instruction word group on a primary storage 20 (VLIW group) are put, the instruction register 12 with which VLIW by which the fetch was carried out from this instruction cache 11 is held, and this instruction register 12 (decoding).
[0029] A decoder 13-1 interprets the contents of a setting of the 1st instruction field in VLIW (part except the instruction validation information a) unconditionally. On the other hand, other decoders 13-2 to 13-4 are the interpretation of the contents of a setting of the 2nd - the 4th instruction field in VLIW Bit a2 -a4 of the instruction validation information a in VLIW It carries out according to a condition (is it "1" or not?).
[0030] It had -4 and CPU10 is equipped with the operand cache 16 on which some counterparts of the operand group on the computing element 14-1 which calculates according to the interpretation result of a decoder 13-1 to 13-4 - the 14 primary storage 20 are put with an instruction execution and the activation / write-in section 15 which manages the writing of an activation result again. The cache line length of this operand cache 16 shall be the same (here 128 bits) with VLIW length. and ** arranged at a 128-bit aryne, without the VLIW straddling two or more cache lines when VLIW for which some instruction fields are used as a variable/a constant area is held as operand data at the operand cache 16 concerned -- ** -- it carries out.
[0031] CPU10 is further equipped with the memory management section 17 which manages page management of a primary storage 20, address translation (from the logical address to a physical address), etc. The memory management section 17 is equipped with TLB (translation lookaside buffer) 171 which is a translation lookaside buffer device for changing the logical address into a high speed in a physical address, and the access-control circuit 172 which performs control of an access privilege etc.
[0032] TLB171 has two or more entries for holding the access privilege information for memory protection, the tag information (address tag) generated based on the logical address (here logic page address), and information including the physical address (here physical page address) corresponding to the logical address concerned. Access privilege information shows either access privilege RO which permits only a lead to a corresponding page, the access privilege RW which permits both a lead and a light access privilege E, etc. The corresponding page is used as a storing field of VLIW, and access privilege E shows "activation being good", if there is no instruction effective in the field in VLIW used as an access place. [that the field can be accessed like the usual operand access]
[0033] A primary storage 20 is managed by the memory management section 17 per page. The field in a primary storage 20 is divided into the page in which VLIW is stored, the page in which operand data is stored. However, in this example, VLIW which contains an operand so that it may mention later exists.
[0034] Next, actuation of the 1st example of this invention is explained. First, the instruction fetch demand should occur in CPU10. At the time of the cache hit in which the demanded instruction word (VLIW) exists in an instruction cache 11, the instruction word (VLIW) is read into an instruction register 12 from an instruction cache 11 by the high speed.
[0035] At the time of the mistake hit in which the demanded instruction word (VLIW) does not exist in an instruction cache 11 on the other hand, after the fetch of the instruction word (VLIW) is carried out from a primary storage 20 through the memory management section 17 and it is held at an instruction

cache 11, the fetch of it is carried out to an instruction register 12 from the instruction cache 11 concerned.

[0036] In addition, about the judgment algorithm of a cache hits / misses hit, and registration processing of the instruction word (VLIW) to an instruction cache 11, it is well known from the former, and since it moreover is not directly related to this invention, explanation is omitted.

[0037] Bit a2 -a4 of the triplet (set as high order part of 1st instruction field) instruction validation information a in VLIW read into the instruction register 12 It is led to a decoder 13-2 to 13-4. Moreover, the contents of a setting of the 1st instruction field in the VLIW concerned (part except the instruction validation information a) (instruction #1) are led to a decoder 13-1, and the contents of a setting of the 2nd instruction field (instruction #2, or a variable/constant) are led to a decoder 13-2. Furthermore, the contents of a setting of the 3rd instruction field in the VLIW concerned (instruction #3, or a variable/constant) are led to a decoder 13-3, and the contents of a setting of the 4th instruction field (instruction #4, or a variable/constant) are led to a decoder 13-4.

[0038] A decoder 13-1 interprets unconditionally the contents of a setting of the 1st instruction field of VLIW drawn from the instruction register 12 (instruction #1) (decoding), and controls the computing element 14-1 of the activation / write-in section 15 according to the interpretation result.

[0039] On the other hand, a decoder 13-2 to 13-4 is bit a2 -a4 of the instruction validation information a drawn from the instruction register 12. It responds and operates as follows. First, bit a2 -a4 of the instruction validation information a In being "1" A decoder 13-2 to 13-4 as that by which the effective required instruction (instruction #2-#4) of an interpretation and activation is set as the 2nd - the 4th instruction field of VLIW drawn from the instruction register 12 The contents of a setting of the instruction field (instruction #2-#4) are interpreted (decoding), and the computing element 14-2 to 14-4 of the activation / write-in section 15 is controlled according to the interpretation result.

[0040] Next, bit a2 -a4 of the instruction validation information a In being "0" An effective required instruction of an interpretation and activation is not set to the 2nd - the 4th instruction field of VLIW to which the decoder 13-2 to 13-4 was led from the instruction register 12. As that to which the variable/constant may be set instead, the instruction field is treated as an invalid (an invalid instruction field, unnecessary instruction field), it refrains from the interpretation of the contents of a setting of the instruction field, and a computing element 14-2 to 14-4 is not operated. In addition, it is available for it that the interpretation result is disregarded even if it carries out, even if it makes it the interpretation (decoding) actuation itself not operate a corresponding calculating machine.

[0041] Bit a2 -a4 of the instruction validation information a prepared in a part of VLIW (here head part of the 1st instruction field) by actuation of the above decoder 13-2 to 13-4 If nullification of an instruction is specified In the 2nd - the 4th instruction field to which VLIW corresponds, no matter what data [32 / hit] it may arrange, the computing element 14-2 to 14-4 corresponding to the field does not malfunction.

[0042] Therefore, if it is the former, the instruction field in VLIW with the need of arranging a nop instruction can be used as a setting field (a variable/constant area) of a variable or a constant. However, the contents of a setting of the 1st instruction field are always interpreted by the decoder 13-1, and since it performs, when activation is unnecessary, they need to arrange a nop instruction as usual in the field concerned.

[0043] In addition, in this example, the instruction length arranged in one VLIW has 29 bits and two kinds of 32 bits (refer to drawing 2). However, it is easy, as it does not become a problem from the VLIW itself being a fixed length, the instruction (instruction #1) of 29 bit length always being set as the 1st instruction field of VLIW moreover, and the instruction (instruction #2-#4) of 32 bit length always being set as the 2nd - the 4th instruction field of VLIW at all but the hardware configuration was also shown in drawing 1 (differing from the method which deletes a nop instruction and makes VLIW variable length). Moreover, although the class of instruction which can be executed with the computing element 14-1 corresponding to the 1st instruction field of VLIW becomes less than the class of instruction which can be executed with the computing element 14-2 to 14-4 corresponding to other instruction fields, it can cope with it by arranging the instruction which cannot be executed with a

computing element 14-1 to the instruction field 2nd after VLIW, and making it perform with a computing element 14-2 to 14-4.

[0044] Now, the interpretation of an instruction is performed by at least one of the decoders 13-1 to 13-4, and it is necessary to lead an operand (a variable/constant) required for the actuation in case the computing element with which it corresponds in the activation / write-in section 15 operates or (from an operand cache 16), and after actuation of a computing element is performed, it is necessary to carry out the light of the result of an operation (to for example, operand cache 16). In this case, an operand access (operand data access) demand occurs from the activation / write-in section 15 to the memory management section 17.

[0045] Then, in the memory management section 17, operand access processing described below according to the flow chart of drawing 3 is performed. First, the access-control circuit 172 in the memory management section 17 will search the entry into which the physical page address corresponding to the address (logic page address) of the demand place is registered with reference to TLB171, if the operand access request from the activation / write-in section 15 is received.

[0046] If it is at the hit time in which the target entry exists in TLB171, the access-control circuit 172 will acquire the access privilege information in the entry concerned while performing address translation of the common knowledge to a physical address from the logical address at a high speed using the physical page address in the entry concerned.

[0047] On the other hand, if it is at the mistake hit time in which the target entry does not exist in TLB171, the access-control circuit 172 will access the page table (not shown) put on the predetermined field of a primary storage 20, and will perform address translation of the common knowledge to a physical address from the logical address using the page table concerned. The access privilege information for every page is set to this page table by management of OS (operating system). Access privilege E peculiar to this example other than the access privilege for the memory protection well known from the former, such as the access privilege RW which permits both access privilege RO which permits only a lead, a lead, and a light as this access privilege information, is prepared. said corresponding page is used as a storing field of VLIW, and this access privilege E shows "activation being good", if it becomes without an instruction effective in the field in VLIW used as an access place, as carried out. [that that field can be accessed like the usual operand access] In this example, it is RO=0, RW=1, and E= 2.

[0048] The access-control circuit 172 will perform registration actuation of the common knowledge which registers into the entry in TLB171 the access privilege information on the applicable page obtained from the table concerned, the tag information (address tag) generated based on the logical address (here logic page address) of a demand place, and the information containing the physical page address which it is as a result of address translation, if address translation which used the page table is performed.

[0049] The access-control circuit 172 by now, the address translation which used TLB171 or a page table If it is at the lead time of an operand when the physical address and access privilege information corresponding to the demand place address (logical address) are acquired (step S1) If it is at the light time of an operand when the access privilege which access privilege information shows is RO or RW, when the access privilege which access privilege information shows is RW Operand access for an operand cache 16 based on the acquired physical address is performed, and the operand led when it was at the lead time is passed to the activation / write-in section 15. The actuation in this case is not different from the former at all. In addition, it cannot be overemphasized that a primary storage 20 is accessed at the time of the mistake hit in which the target data do not exist in an operand cache 16.

[0050] On the other hand, when the access privilege which access privilege information shows is E (i.e., when activation is good), (step S2) and the access-control circuit 172 carry out read access of the cache line where an operand cache 16 corresponds based on the acquired physical address (step S3). When an access privilege is E (activation is possible), the information on the accessed cache line is VLIW.

[0051] The access-control circuit 172 takes out the instruction validation information a from the 1st instruction field of 32 bits of the head of the information (VLIW) on the accessed cache line, i.e., VLIW,

(step S4). In addition, at the time of a mistake hit, ejection of the instruction validation information a is performed for a primary storage 20.

[0052] The access-control circuit 172 will investigate whether an instruction effective in the field (data word part) which should carry out operand access exists based on the correspondence bit (condition) of the instruction validation information a concerned, if ejection of the instruction validation information a is performed (step S5).

[0053] Supposing an effective instruction exists, the access-control circuit 172 will protect the instruction concerned, without performing demanded operand access as unlawful access (access error), and will notify that to an operand cache 16 (step S6).

[0054] On the other hand, if it becomes, the access-control circuit 172 will perform access for the field (data word part) where an effective instruction does not exist and which should carry out operand access like the usual operand data (step S7).

[0055] Thus, the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW can be reduced by using the instruction field which must arrange a nop instruction as a variable/a constant area, if it is the former in the dependency of an instruction among the 2nd - the 4th instruction fields among VLIW. And since nullification treatment of the contents of a setting of the instruction field used as a variable/a constant area is interpreted and carried out by nullification assignment of the correspondence bit of the instruction validation information a by corresponding decoder 13-i (i is either 2-4), there is no possibility that computing-element 14-i may malfunction.

[0056] Drawing 4 shows the configuration of the compiler which generates the object program containing the VLIW group of a format as shown in drawing 2 applied with the computer system of drawing 1 from a source program.

[0057] The compiler of drawing 4 consists of the 1st compile section 41 which generates the 1st object program 32 which compiles a source program 31 and contains the VLIW group of the same format as usual, and the 2nd compile section 42 which generates the 2nd object program 33 of the format of drawing 2 from this 1st object program 32.

[0058] Next, actuation of the compiler of the configuration of drawing 4 is explained with reference to the flow chart of drawing 5 and drawing 6 . First, from the source code train of a source program 31, for every source code, the 1st compile section 41 creates an instruction or instruction group of an activation mold serially, and generates the instruction train before the relocation to VLIW (object code train) (step S11).

[0059] Next, the 1st compile section 41 optimizes and rearranges the instruction train before the generated relocation in a VLIW mold with four 32-bit instruction fields, it is arranging a nop instruction to the instruction field which cannot arrange the instruction which operates a computing element according to the dependency during each instruction, and the 1st object program 32 containing the VLIW group of the same format as usual is generated (step S12). However, in the 1st instruction field of VLIW, it differs from the former for a while in that a 29-bit instruction or a nop instruction is arranged to the field except a top triplet. In addition, the 29-bit instruction is equivalent to the 32-bit instruction in which operation assignment is possible regardless of the triplet of a high order.

[0060] The 1st object program 32 generated by the 1st compile section 41 as mentioned above consists of a code section 321 which consists of a VLIW group (the same format as usual), and data division 322 which are the sets of the data (a variable/constant) referred to from the instruction on a code section 321. The data of these data division 322 are related with the information (label) which shows the logical location on which that data is put.

[0061] The 1st compile section's 41 generation of the 1st object program 32 starts the 2nd compile section 42. Then, the 2nd compile section 42 confirms whether the nop instruction is arranged there with reference to the 2nd instruction field of the head VLIW of the code section 321 of the 1st object program 32 (step S13) (step S14).

[0062] If the 2nd compile section 42 uses the field as a variable/a constant area when the instruction field by which the nop instruction is arranged is detected (step S15), it will arrange the data of the

arbitration on the data division 322 of the 1st object program 32 in the field (step S16). (replacing with a nop instruction)
[0063] Next, the 2nd compile section 42 looks for all instructions that refer to the data which have scanned and arranged the code section 321, and rewrites them, respectively on the label in which the location of the instruction field which has arranged the label in which the reference place of the data concerned set as each [these] instruction is shown at step S16 is shown (step S17). In addition, if the positional information of the instruction which refers to the data on data division 322 among each instruction on a code section 321 is made to correspond with the data concerned and is held on the table etc. in case the 1st compile section 41 generates the 1st object program 32, the instruction at the above-mentioned step S17 set as the object of label rewriting can be looked for easily.
[0064] The 2nd compile section 42 will confirm whether it ended to the instruction field of the last of VLIW (the present VLIW) which is carrying out current attention, if step S17 is performed (step S18). The check of this step S18 is performed, when the nop instruction is not arranged at the instruction field referred to, and also when not using the instruction field referred to as a variable/a constant area.
[0065] If it has not ended to the instruction field of the last of the present VLIW, the 2nd compile section 42 returns to step S14 with reference to the next instruction field of the present VLIW (step S19).
[0066] On the other hand, if it has ended to the instruction field of the last of the present VLIW, the 2nd compile section 42 will set the instruction validation information a according to the contents of the 2nd - the 4th instruction field as the high order triplet part of the 1st instruction field of the VLIW (step S20).
[0067] The 2nd compile section 42 will confirm whether it ended to VLIW of the last of a code section 321, if setting processing of the instruction validation information a is performed (step S21).
[0068] If it has not ended to VLIW of the last of a code section 321, the 2nd compile section 42 returns to step S14 with reference to the 2nd instruction field of the next VLIW on a code section 321 (step S22).
[0069] On the other hand, if it has ended to VLIW of the last of a code section 321, the 2nd compile section 42 will end compile processing. The 2nd object program 33 which consists of a code section 33 which consists of a group of VLIW of the format of drawing 2 , and data division 322 which are the sets of the data (a variable/constant) referred to from the instruction on a code section 321 by this is generated. This 2nd object program 33 is used with the computing system of drawing 1 .
[0070] In the above, when it was the former, it explained per [aiming at reduction of the fields of the primary storage 20 which an activation object uses] 1st example by enabling it to use the instruction field in VLIW with the need of arranging a nop instruction, as a variable/a constant area.
[0071] In addition, although a part of 1st 32-bit long-lived ** field of VLIW of 128 bit length is used as instruction validation information a, you may make it use VLIW of the 131 bit length which consists of the instruction field and the instruction validation information a on four 32 bit length in the 1st example of the above. In any case, a change is not in using a part of VLIW as instruction validation information a. Moreover, of course, it is not what also restricts the die length of an instruction field, and the number of instruction fields to the 1st example of the above.
[0072] Moreover, although the 1st object program 32 including the train of VLIW of the same format as usual shall be generated in step S12 of the compile processing according to the flow chart of drawing 4 and step S20 shall perform a setup of the instruction validation information a in the 1st example of the above A meaningless code different from a nop instruction is arranged to an invalid instruction field, or it is vacant, the field concerned is made into the field, and you may make it arrange the instruction validation information a to the head instruction field of VLIW in the phase of step S12.
With reference to a drawing, it explains about the 2nd example aiming at reduction of the auxiliary storage areas (disk field) which the file concerned occupies by the ability to be made to make small size of the load module file of the [2nd example], next a VLIW method.
[0073] Drawing 7 generates a load module and stores it in secondary memory, and when a TEMANDO page access request occurs, the configuration for developing the page of the demanded load module to a primary storage is shown.

[0074] As for the linkage editor as a link means, and 52, in this drawing, 51 is [the virtual-memory-management section of an operating system (OS) and 53] primary storages. A linkage editor 51 links the object group (object program group) 61 of a VLIW method, and generates the compressed load module 62 which can be performed with a computer system (inner CPU). The object group 61 applied by this example shall consist of trains of VLIW of the format of drawing 2 applied in said 1st example, i.e., four 4-byte instruction fields, (VLIW length = 16 bytes), and shall consist of a train of VLIW which has the instruction validation information a in the head part of the 1st instruction field. However, unlike said 1st example, the instruction field in which invalid (unnecessary) assignment is carried out by the bit ai (i=2-4) within the instruction validation information a (as a thing without the need for an interpretation and activation) shall not be used as a variable/a constant area, but shall be positioned as a free area.

[0075] A load module 62 is stored in the load module file 63 put on secondary memory, for example, a disk unit, and consists of a code section 621 for pagination, and an index part 622 for pagination.

[0076] Each code section 621 is used as a compressed code field where the instruction group (the remaining instruction group by which the invalid instruction-field part was specifically deleted from the VLIW train) which compressed the VLIW train of a corresponding page is stored.

[0077] Each index part 622 is for storing the information (a page index being called hereafter) which shows the head location of the corresponding code section 621 of a page (in relative position in a load module file 63).

[0078] When a demand page access request occurs, the virtual-memory-management section 52 restores the VLIW train of a page from the load module 62 in a load module file 63, and arranges it to a primary storage 53.

[0079] Next, actuation of the configuration of drawing 7 is divided into (1) link processing and (2) demand-page access processing, and is explained in order with reference to the flow chart of drawing 8 thru/or drawing 10 .

(1) link **** -- explain the link processing by the linkage editor 51 first.

[0080] In case a linkage editor 51 links the object group 61 and a load module 62 is generated, the index part 622 for the pagination to need is secured in a load module file 62 (step S31).

[0081] Next, a linkage editor 51 secures the code section 621 for the head pages of a load module 62 in a load module file 62, and sets the page index which shows the load module file 63 internal-phase pair location of the head of the code section 621 as an object page at the index part 622 of a proper (step S32).

[0082] Next, a linkage editor 51 takes out the head VLIW of an object page from the object group 61 (step S33). And a linkage editor 51 removes the instruction-field part it is indicated to be that it is the invalid (needlessness) field without the need for an interpretation and activation from the VLIW concerned with reference to the instruction validation information a set as the head part of the 1st instruction field of taken-out VLIW, and stores it in a code section 621 for example, per 1 instruction field (step S34).

[0083] Thereby, when the instruction validation information a (inner bit a2 -a4) is VLIW which is "010", the 2nd and the 4th instruction-field part in the VLIW concerned are deleted, and only the contents of the 1st and the 3rd remaining instruction field are stored in a code section 621. Similarly, when the instruction validation information a (inner bit a2 -a4) is VLIW which is "101", the 3rd instruction-field part in the VLIW concerned is deleted, and only the contents of the 1st, the 2nd, and the 4th remaining instruction field are stored in a code section 621. This situation is shown in drawing 11 (a).

[0084] Next, it is checked whether it ended to VLIW of the last of an object page, and a page end for a linkage editor 51 (step S35). Here, when a page size is made into 16 K bytes, since 1VLIW consists of four 4-byte instruction fields, the number of VLIW of (VLIW length =16 byte) and 1 page is 1024. Therefore, whenever a linkage editor 51 processes 1024 VLIW, it judges a page end.

[0085] If it is not a page end, a linkage editor 51 will take out the next VLIW of the same page from the object group 61 (step S36), and will return to step S34.

[0086] On the other hand, if it is a page end, a linkage editor 51 will confirm whether processing for required pagination was ended (step S37). If it becomes, a linkage editor 51 will secure the code section 621 for the following pages in a load module file 62, and will set the page index which has not ended processing for required pagination and which shows the load module file 63 internal-phase pair location of the head of the code section 621 as an object page at the index part 622 of a proper (step S38). And a linkage editor 51 returns to step S33.
[0087] On the other hand, if processing for required pagination is ended, a linkage editor 51 will end link processing. Thus, the load module 62 which has the code section 621 by which the instruction group (compressed VLIW train) by which the invalid instruction-field part without the need for an interpretation and activation was removed from each VLIW was stored in the load module file 63, and the index part 622 which shows the load module file 63 internal-phase pair location of the head of the code section 621 concerned by pagination is generated. Since this load module 62 does not contain the invalid instruction-field part in VLIW, its size of a load module file 63 is small compared with the former, and it ends.
(2) Explain demand page access processing, next the demand page access processing by the virtual-memory-management section 52.
[0088] If a demand page access request occurs working [a computer system], actuation which arranges the VLIW train of a page which accessed the load module file 63 and was demanded to the appointed field on a primary storage 53 (loading) will be performed as follows by the virtual-memory-management section 52.
[0089] The virtual-memory-management section 52 sets up the head location of the above-mentioned appointed field first as a value of the pointer (arrangement place pointer) indicating the instruction arrangement place on a primary storage 53 (step S41).
[0090] Next, the virtual-memory-management section 52 recognizes the head location of the code section (the instruction group is stored) 621 of the page concerned with reference to the index part 622 of a proper to the page demanded in the load module 62 (step S42).
[0091] Next, the virtual-memory-management section 52 takes out a top instruction, i.e., the contents of the 1st field of VLIW, from the code section 621 (step S43). Next, the virtual-memory-management section 52 is arranged to the 4-byte field of a primary storage 53 to which an arrangement place pointer points out the taken-out instruction (the contents of the 1st field of VLIW) (loading) (step S44), and carries forward the pointer concerned by 4 bytes (step S45).
[0092] next, the virtual-memory-management section 52 -- bit ai in the instruction validation information a the picking appearance above-mentioned after setting the variable i for specifying as initial value 2 (step S46) -- bit ai in the instruction validation information a set as the head part of an instruction the bottom referring to (step S47) -- the ai concerned It confirms whether be "1" or not (step S48).
[0093] If it is ai =1, the virtual-memory-management section 52 takes out the next instruction from a code section 621 (step S49), and it arranges to the 4-byte field of a primary storage 53 which an arrangement place pointer points out (step S50), and while carrying forward the pointer concerned to after an appropriate time by 4 bytes, i will be carried out +one (steps S51 and S52).
[0094] On the other hand, if it is ai =0, the virtual-memory-management section 52 skips steps S49 and S50, and while carrying forward an arrangement place pointer by 4 bytes, i will be carried out +one (steps S51 and S52). By this, the field of an invalid instruction field will be secured on a primary storage 53.
[0095] The virtual-memory-management section 52 will confirm whether the value of i exceeded 4, if steps S51 and S52 are performed (step S53). If i has not exceeded 4, the virtual-memory-management section 52 will return to step S47, and will perform steps S48-S53 or steps S48, S51, and S53 with the value with reference to the next bit in the instruction validation information a (bit ai).
[0096] On the other hand, if i has exceeded 4, the virtual-memory-management section 52 would judge it as what 1VLIW has restored and arranged on a primary storage 53, and it will be checked whether it has arranged to VLIW of the last of the demanded page, and a page end for it (step S54).

[0097] if -- a page -- and -- ** -- it has not become -- if it becomes, the virtual-memory-management section 52 will take out the contents of the 1st field of the next instruction, i.e., the following VLIW, from a code section 621 (step S55), and will return to step S44.
[0098] On the other hand, if it is a page end, the virtual-memory-management section 52 will end demand page access processing as what all the demanded VLIW trains of a page have restored and arranged at the primary storage 53.
[0099] According to the above demand page access processing, according to the instruction validation information a in the head part of the instruction (the contents of the 1st instruction field of VLIW) taken out at step S43 or step S55, the original VLIW is correctly restored from the contents of the code section 621 in which the contents of the invalid field of VLIW are deleted and stored, and it is arranged at a primary storage 53 (loading).
[0100] for example, when the instruction validation information a in the head part of the instruction taken out at step S43 or step S55 (inner bit a2 -a4) is "010" Where it skipped by one instruction from the instruction (instruction #1) concerned on the primary storage 53 and an invalid instruction field is secured The next instruction on a code section 621 (instruction #3) is arranged, and the original VLIW is restored and arranged by skipping to after an appropriate time by further 1 instruction, and securing an invalid instruction field. Similarly, when the instruction validation information a in the head part of a head instruction of VLIW (inner bit a2 -a4) is "101" Where it has arranged the next instruction on a code section 621 (instruction #2), it skipped to after an appropriate time by one instruction and an invalid instruction field is secured in the consecutiveness location of the instruction (instruction #1) concerned on a primary storage 53 The original VLIW is restored and arranged by the thing on a code section 621 for which the next instruction (instruction #4) is arranged further. This situation is shown in drawing 11 (b).
[0101] In the above, the auxiliary storage areas (disk field) which a load module file occupies could be reduced by generating the load module with which a part for invalid (unnecessary) instruction part (instruction field) was removed at the time of link processing, and it explained per [which enabled it to perform normal program manipulation by restoring to the original VLIW train at the time of the program load which moreover follows a demand page access request] 2nd example.
With reference to drawing 12 , it explains about the 3rd example applied to the computing system treating the train of VLIW by which a nop instruction is set as the train, i.e., the invalid instruction field, of VLIW of the same format as usual in the technique which reduces the load module size stated in the [3rd example], next said 2nd example.
[0102] As for a linkage editor and 72, in drawing 12 , 71 is [the virtual-memory-management section and 73] primary storages. A linkage editor 71 links the object group (object program group) 81 of a VLIW method, and generates the compressed load module 82 which can be performed with a computer system (inner CPU). The object group 81 applied by this example shall consist of a VLIW train of the same format as usual.
[0103] A load module 82 is stored in the load module file 83 put on secondary memory, for example, a disk unit, and consists of a code section 821 for pagination, and an index part 822 for pagination.
[0104] Each code section 821 consists of instruction map field 821b in which the map (instruction map) in which arrangement of compressed code field 821a in which the instruction group (the remaining instruction group by which the nop instruction was specifically deleted from the VLIW train) which compressed the VLIW train of a corresponding page is stored, the nop instruction in the VLIW train before compression, and the other instruction is shown is stored.
[0105] Each index part 822 is for storing the information (page index) which shows the head location of the corresponding code section 821 of a page (in relative position in a load module 8).
[0106] When a demand page access request occurs, the virtual-memory-management section 72 restores the VLIW train of a page from the load module 82 in a load module file 83, and arranges it to a primary storage 73.
[0107] Next, actuation of the configuration of drawing 12 is explained. First, in case a linkage editor 71 links the object group 81 and generates a load module 82, it secures the code section 821 in the load

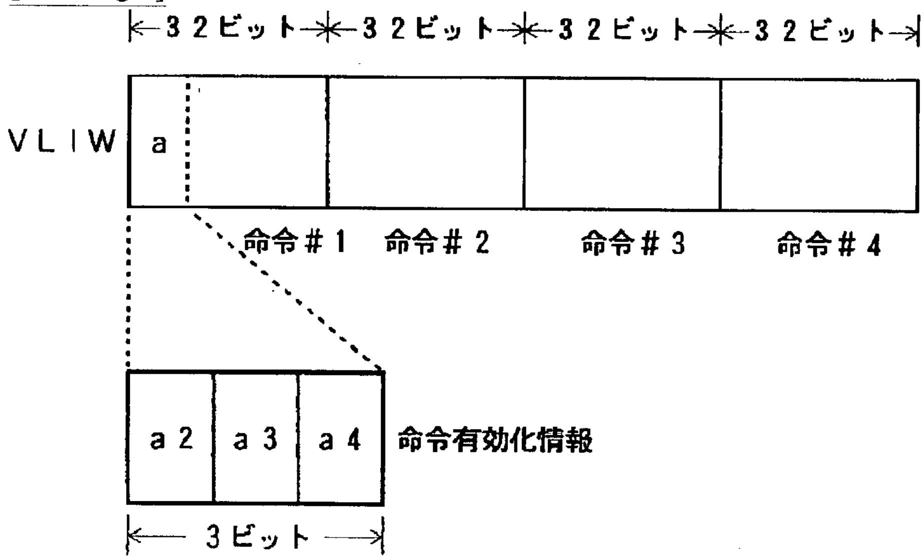
module file 83 assigned to the page sequentially from a head page.
[0108] The linkage editor 71 is stored about the VLIW train of the page in the object group 81 from the head of compressed code field 821a of the code section 821 concerned, removing a nop instruction sequentially from Head VLIW, whenever it secures the code section 821 for 1 page.
[0109] The linkage editor 71 is stored in instruction map field 821b in which the information which shows arrangement of the nop instruction on VLIW and the other instruction sequentially from Head VLIW was prepared at the head of the code section 821 concerned for every VLIW of the page which corresponds again in parallel to instruction storing except the nop instruction to compressed code field 821a of the above-mentioned code section 821.
[0110] the nop instruction on VLIW since the number of VLIW of 1 page is 1024 when 1VLIW consists of four 4-byte instruction fields (VLIW length = 16 bytes) and the size of this instruction map field 821b makes a page size 16 K bytes, and arrangement of the other instruction -- per [one instruction (1 instruction field)] -- 1 bit shows -- if it becomes, it will become the fixed size of 1024x4 bits.
[0111] A linkage editor 71 stores in the target page the page index which shows further the load module file 83 internal-phase pair location of the head of the code section 821 which carried out [above-mentioned] reservation at the index part 822 of a proper.
[0112] Thus, the load module 82 which has the code section 821 by which the instruction group by which the nop instruction was removed was stored in the load module file 83, and the index part 822 which shows the load module file 83 internal-phase pair location of the head of the code section 821 concerned by pagination is generated. Since the nop instruction is removed, the size of a load module file 83 of this load module 82 is small compared with the former, and it ends.
[0113] Now, if a demand page access request occurs working [a computer system], actuation which arranges the VLIW train of a page which accessed the load module file 83 and was demanded to a primary storage 73 will be performed as follows by the virtual-memory-management section 72.
[0114] The virtual-memory-management section 72 recognizes the head location of the code section (the instruction group is stored) 821 of the page concerned with reference to the index part 822 of a proper first to the page demanded in the load module 82.
[0115] Next, referring to the instruction map stored in instruction map field 821b which begins from the head location of the code section 821 sequentially from a head bit, the virtual-memory-management section 72 takes out an instruction sequentially from the head of compressed code field 821a which makes the instruction map field 821b and pair, and arranges it to a primary storage 73 (loading). When the bit referred to is "0" at this time, the virtual-memory-management section 72 judges it as that by which the nop instruction is removed, is preceded with the instruction taken out from compressed code field 821a, and arranges a nop instruction to a primary storage 73. And a nop instruction is followed and the instruction taken out from compressed code field 821a is arranged to a primary storage 73 in the place which changed the bit referred to to "1" from "0."
[0116] Thus, it is restored correctly and the demanded VLIW train of a page is arranged at a primary storage 73. In the above, the auxiliary storage areas (disk field) which a load module file occupies could be reduced by generating the load module with which the nop instruction was removed at the time of link processing, even if it has applied the VLIW train of the same format as usual, and it explained per [which enabled it to perform normal program manipulation by restoring to the original VLIW train at the time of the program load which moreover follows a demand page access request] 3rd example.
[0117]
[Effect of the Invention] As explained in full detail above, the code part which was conventionally occupied with the nop instruction on VLIW according to this invention can be used now as a variable or a constant area, and the main storage areas which an activation object uses by this can be reduced.
[0118] Moreover, the auxiliary storage areas which a load module file occupies can be reduced by generating the load module with which a part for the invalid (unnecessary) instruction part conventionally occupied with the nop instruction on VLIW (instruction field) was removed according to this invention, and normal program manipulation can be performed by restoring to the original VLIW train at the time of the program load which moreover follows a demand page access request. According

to such this invention, the futility of the storage region resulting from the code part conventionally occupied with the nop instruction on VLIW can be reduced.

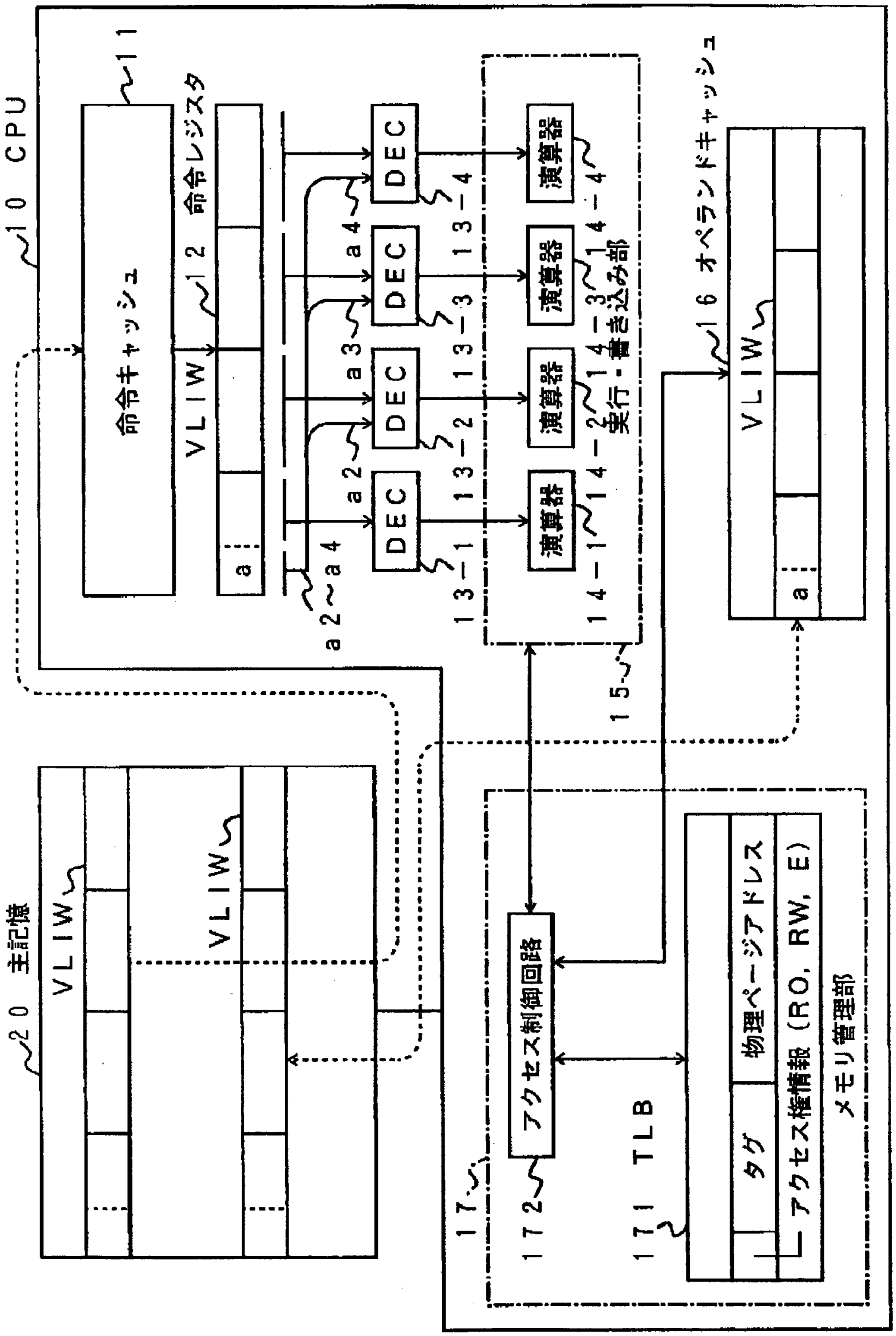
[Translation done.]

DRAWINGS

[Drawing 2]

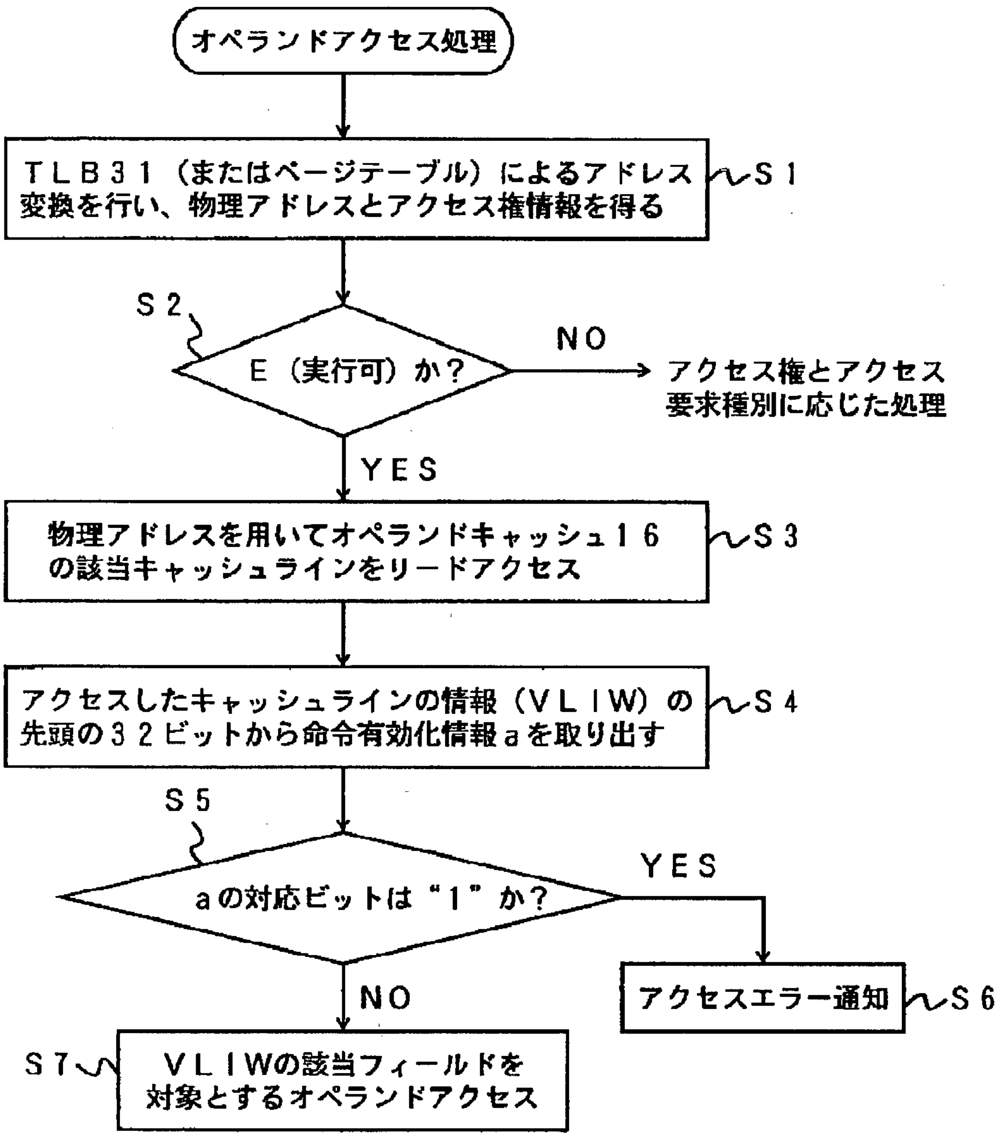


[Drawing 1]

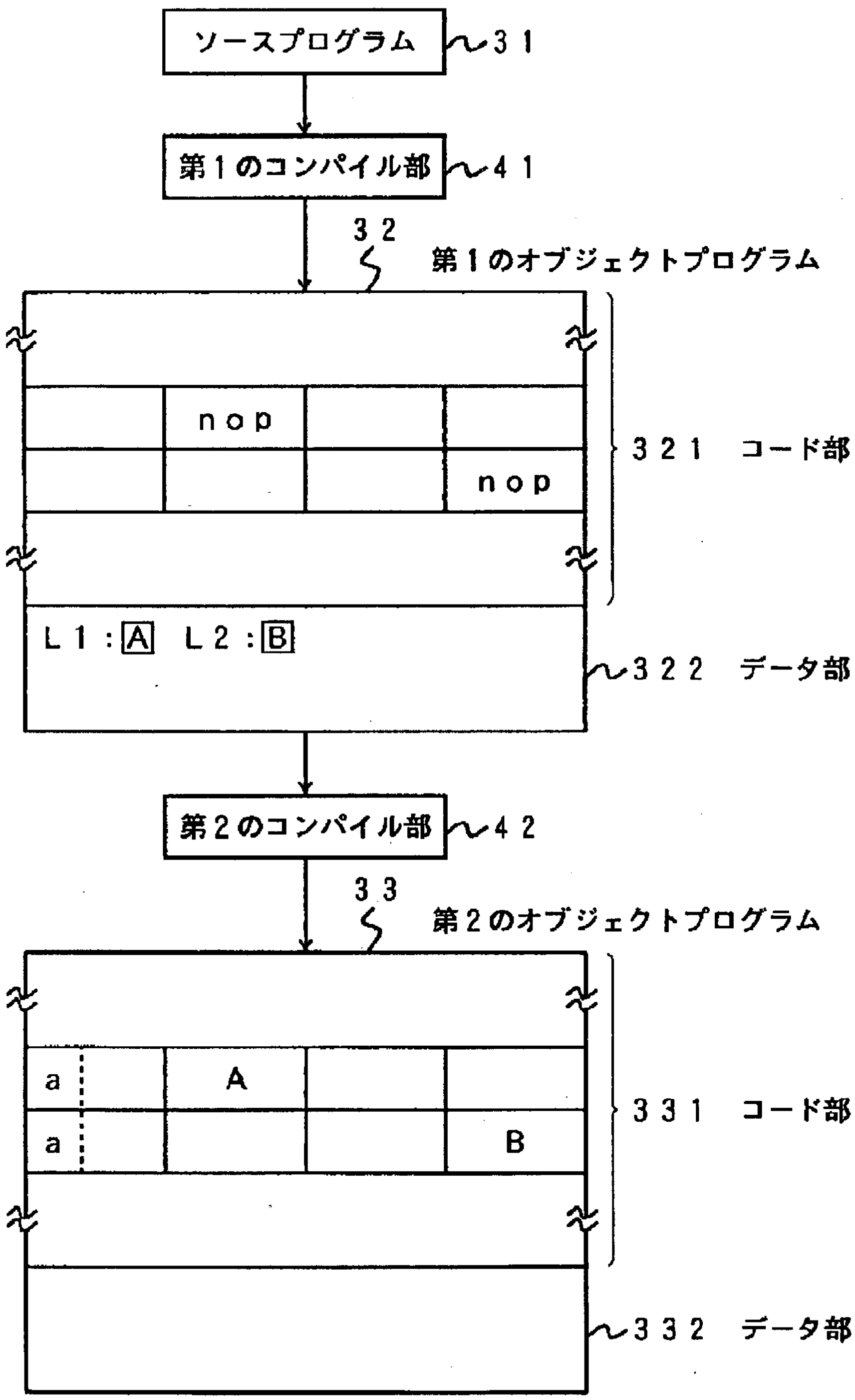


drawing 1

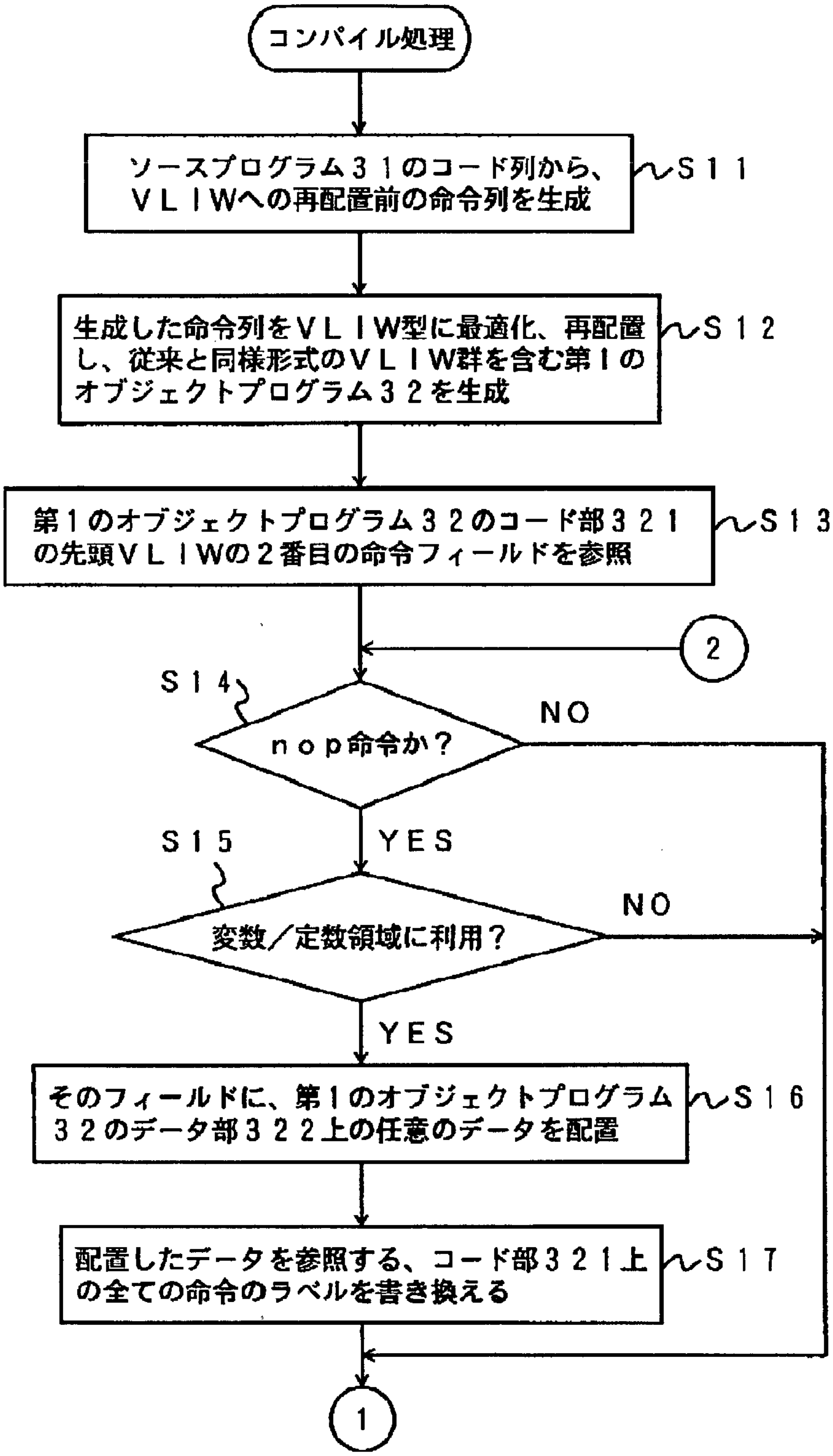
[Drawing 3]


オペランドアクセス処理
ＴＬＢ３１（またはページテーブル）によるアドレス変換を行い、物理アドレスとアクセス権情報を得る
Ｓ１
Ｓ２
Ｅ（実行可）か？
ＮＯ
アクセス権とアクセス要求種別に応じた処理
ＹＥＳ
物理アドレスを用いてオペランドキャッシュ１６の該当キャッシュラインをリードアクセス
Ｓ３
アクセスしたキャッシュラインの情報（ＶＬＩＷ）の先頭の３２ビットから命令有効化情報ａを取り出す
Ｓ４
Ｓ５
ａの対応ビットは“１”か？
ＹＥＳ
アクセスエラー通知
Ｓ６
ＮＯ
Ｓ７
ＶＬＩＷの該当フィールドを対象とするオペランドアクセス

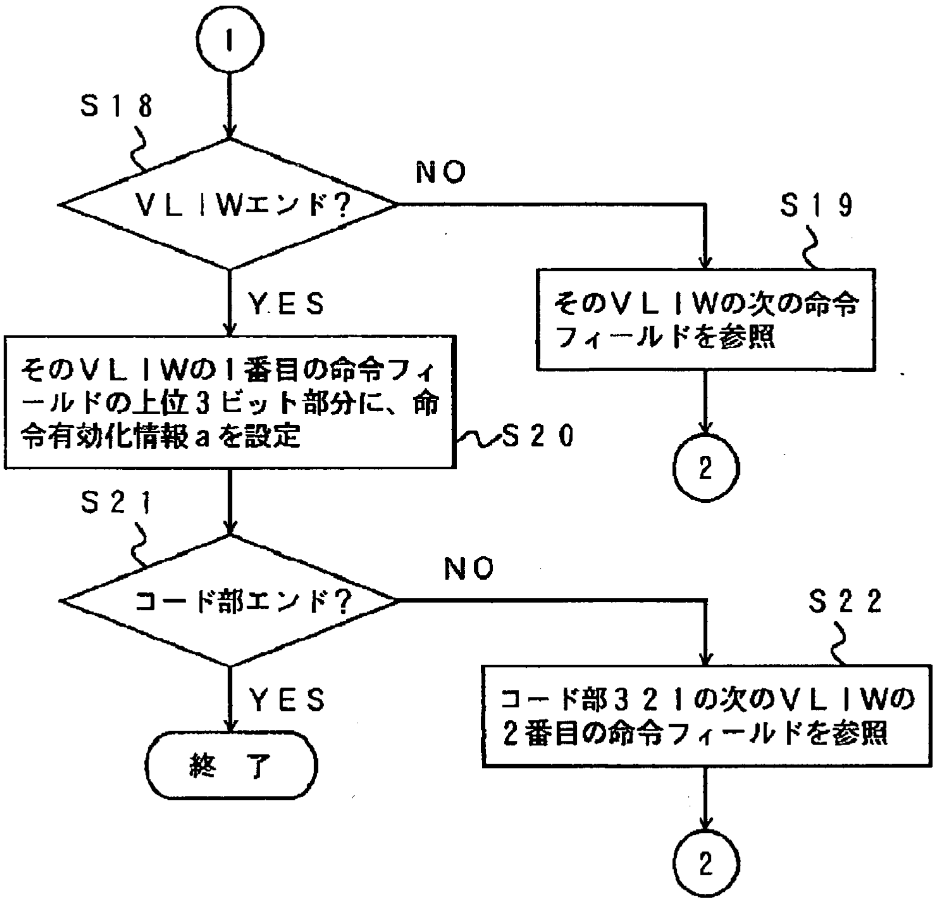
[Drawing 4]


ソースプログラム
31
第1のコンパイル部
41
32
第1のオブジェクトプログラム
nop
nop
321 コード部
L1 : A L2 : B
322 データ部
第2のコンパイル部
42
33
第2のオブジェクトプログラム
a
A
a
B
331 コード部
332 データ部

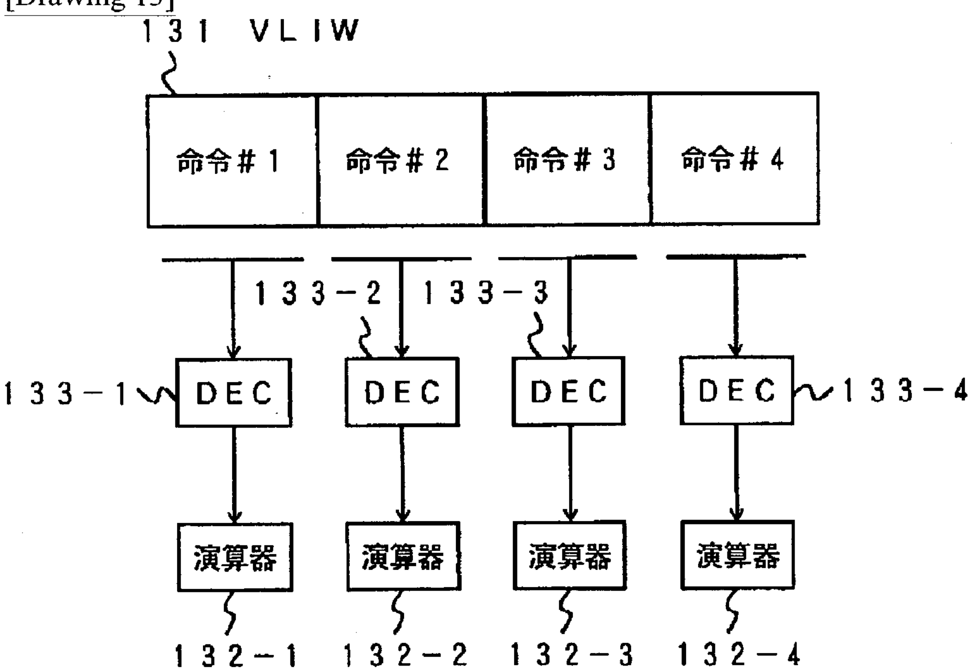
[Drawing 5]


コンパイル処理
ソースプログラム３１のコード列から、ＶＬＩＷへの再配置前の命令列を生成
S11
生成した命令列をＶＬＩＷ型に最適化、再配置し、従来と同様形式のＶＬＩＷ群を含む第１のオブジェクトプログラム３２を生成
S12
第１のオブジェクトプログラム３２のコード部３２１の先頭ＶＬＩＷの２番目の命令フィールドを参照
S13
2
S14
nop命令か？
NO
YES
S15
変数／定数領域に利用？
NO
YES
そのフィールドに、第１のオブジェクトプログラム３２のデータ部３２２上の任意のデータを配置
S16
配置したデータを参照する、コード部３２１上の全ての命令のラベルを書き換える
S17
1

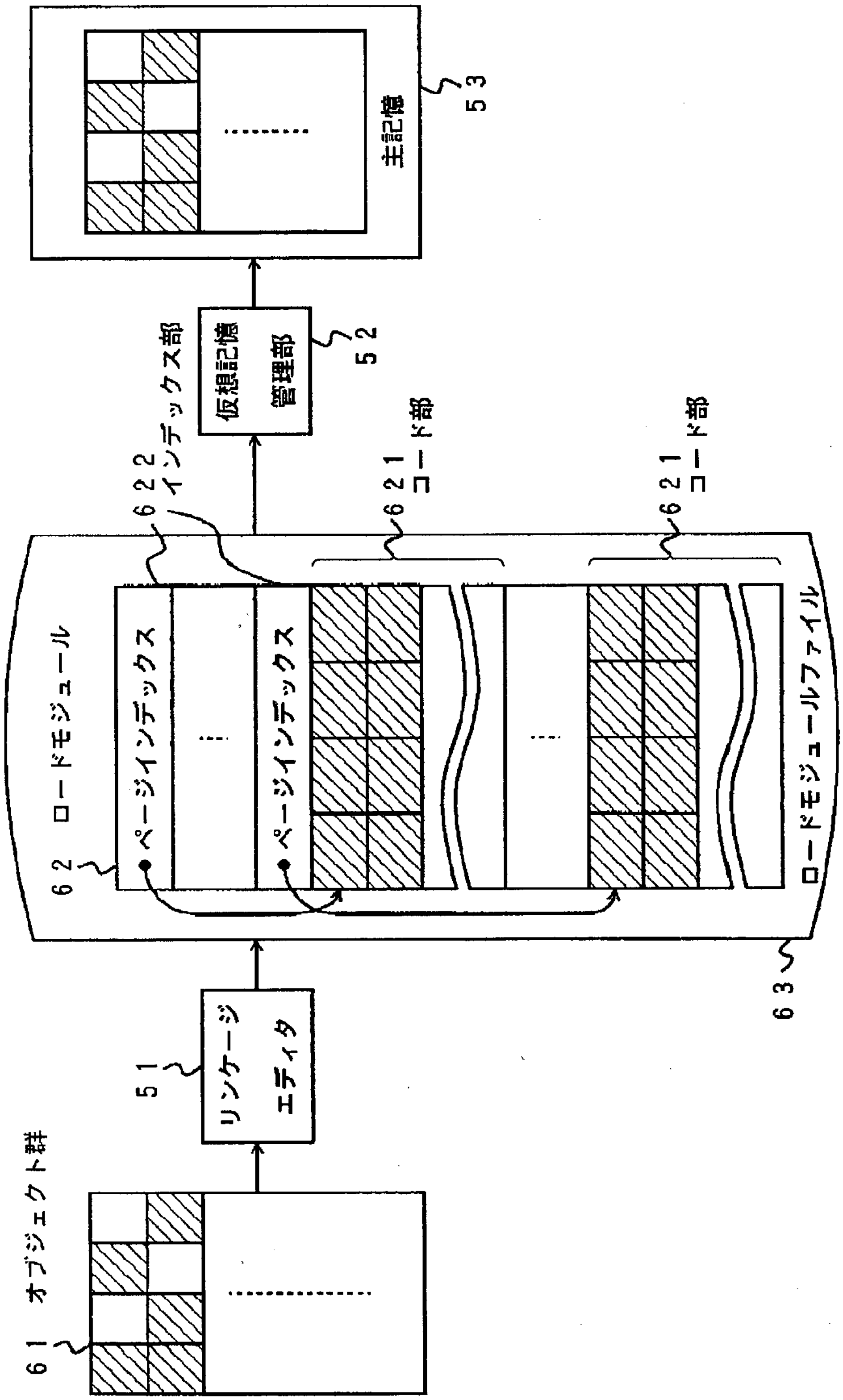
[Drawing 6]



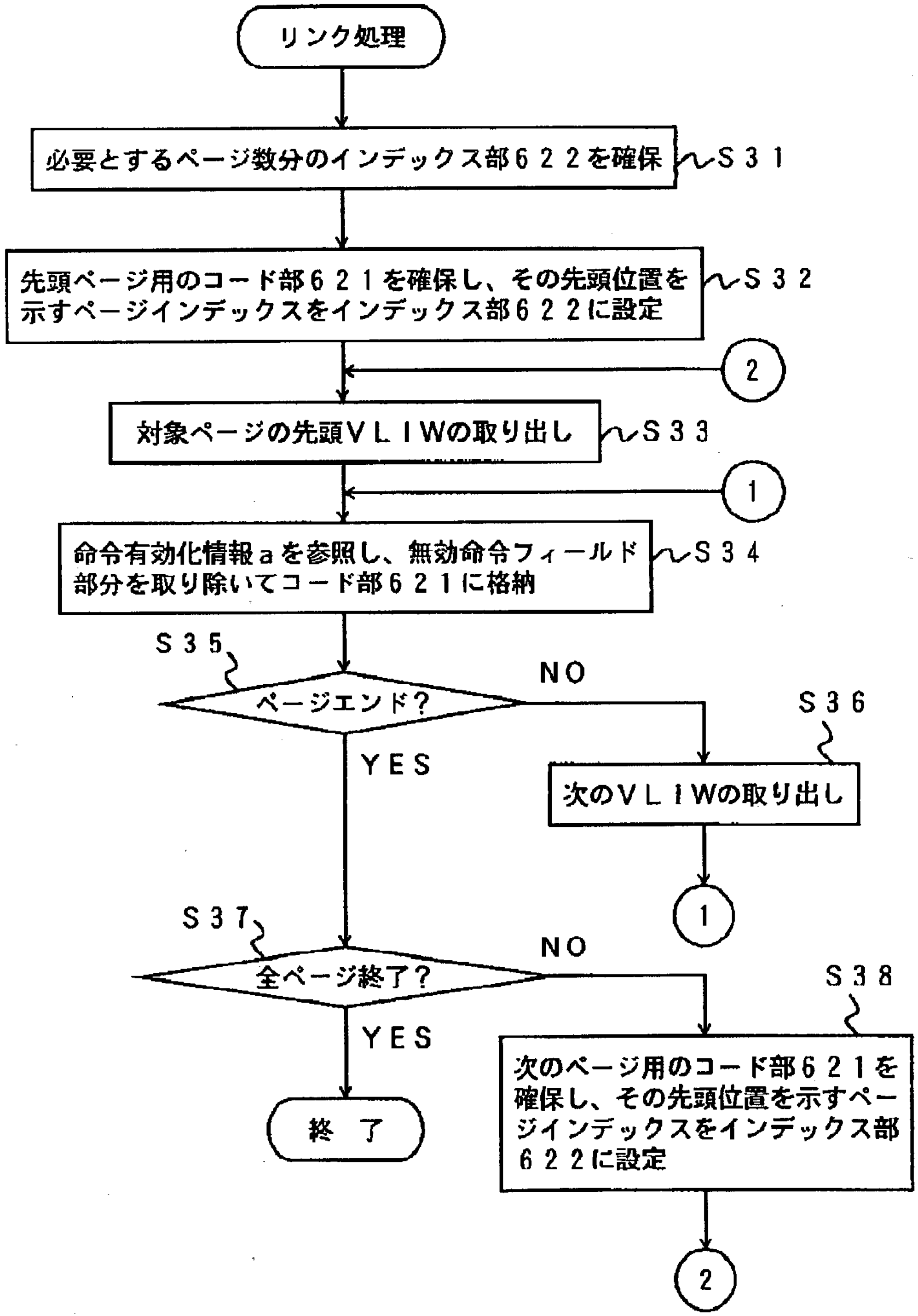
[Drawing 13]



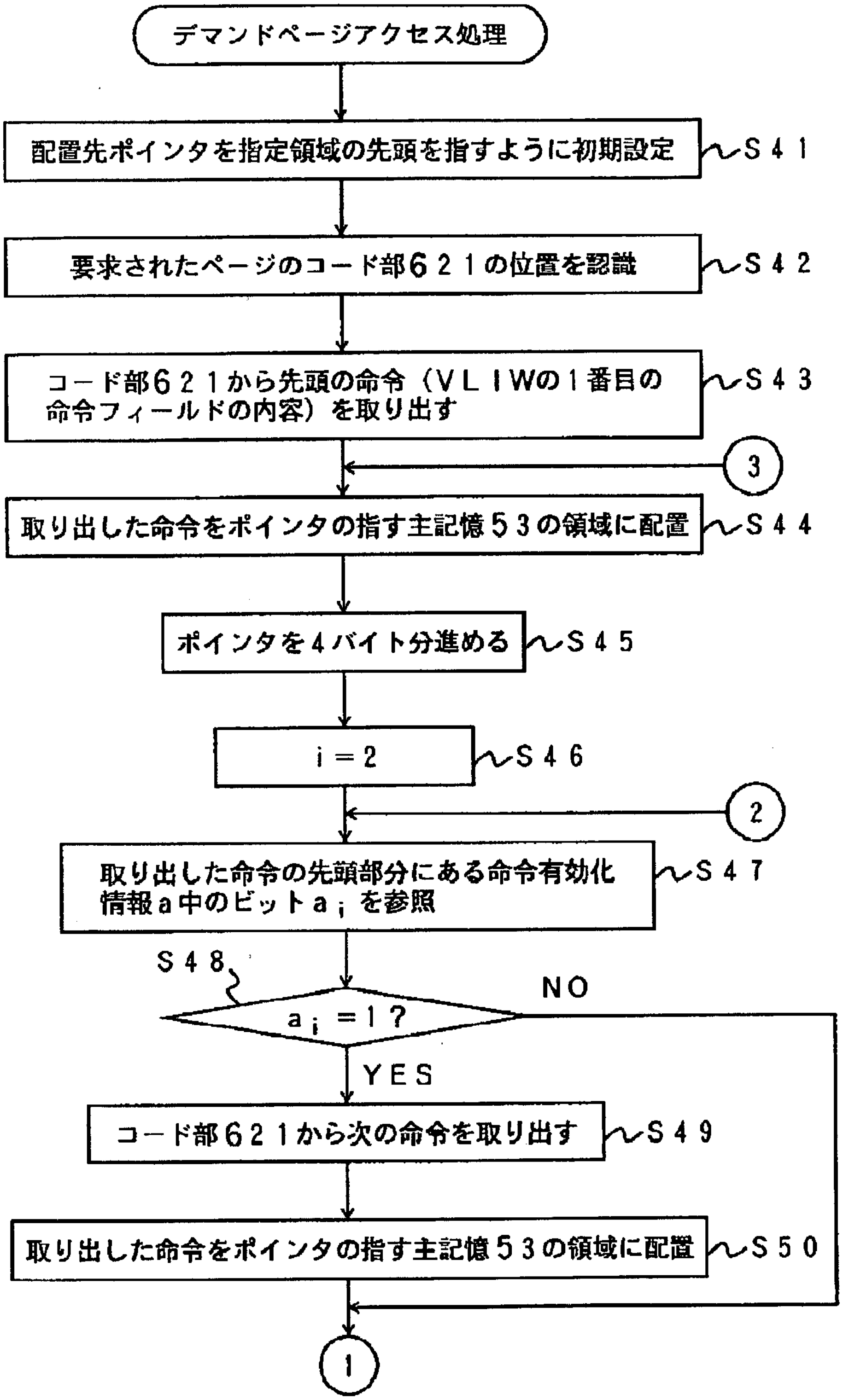
[Drawing 7]


オブジェクト群
61
リンケージエディタ
51
62 ロードモジュール
ページインデックス
ページインデックス
ロードモジュールファイル
63
622
インデックス部
621
コード部
621
コード部
仮想記憶管理部
52
主記憶
53

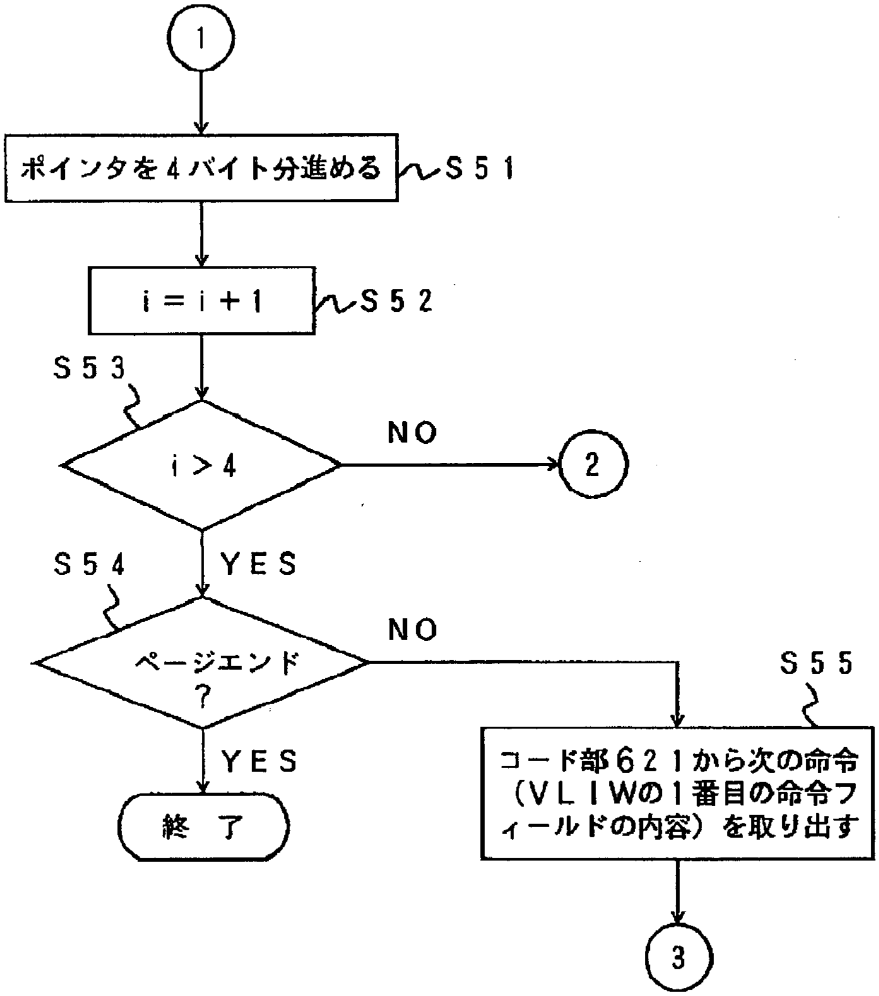
[Drawing 8]


リンク処理
必要とするページ数分のインデックス部６２２を確保
S３１
先頭ページ用のコード部６２１を確保し、その先頭位置を示すページインデックスをインデックス部６２２に設定
S３２
２
対象ページの先頭ＶＬＩＷの取り出し
S３３
１
命令有効化情報ａを参照し、無効命令フィールド部分を取り除いてコード部６２１に格納
S３４
S３５
ページエンド？
NO
YES
S３６
次のＶＬＩＷの取り出し
１
S３７
全ページ終了？
NO
YES
S３８
次のページ用のコード部６２１を確保し、その先頭位置を示すページインデックスをインデックス部６２２に設定
終　了
２

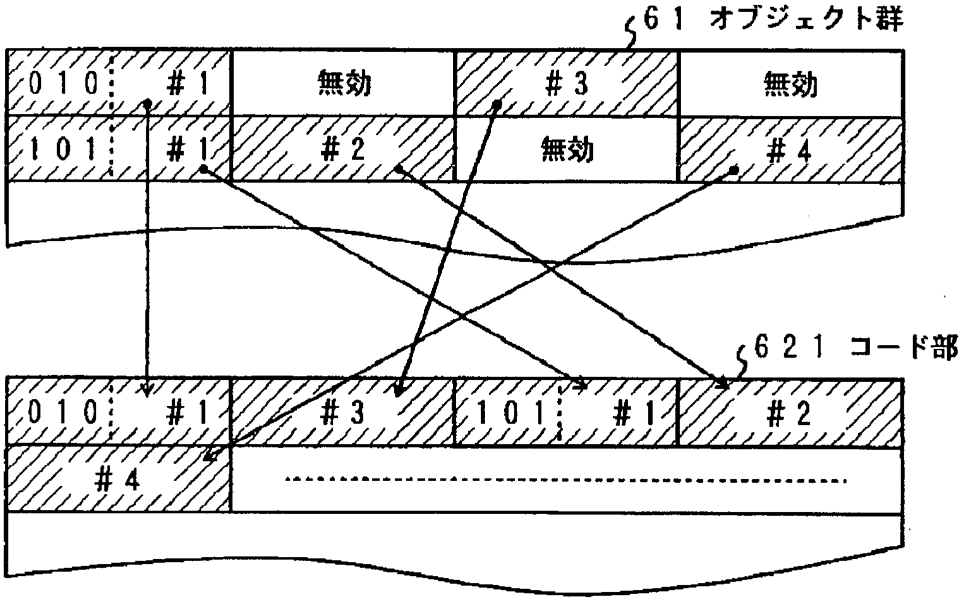
[Drawing 9]


デマンドページアクセス処理
配置先ポインタを指定領域の先頭を指すように初期設定
S41
要求されたページのコード部621の位置を認識
S42
コード部621から先頭の命令（VLIWの1番目の命令フィールドの内容）を取り出す
S43
3
取り出した命令をポインタの指す主記憶53の領域に配置
S44
ポインタを4バイト分進める
S45
i＝2
S46
2
取り出した命令の先頭部分にある命令有効化情報a中のビットa i を参照
S47
S48
a i ＝1？
NO
YES
コード部621から次の命令を取り出す
S49
取り出した命令をポインタの指す主記憶53の領域に配置
S50
1

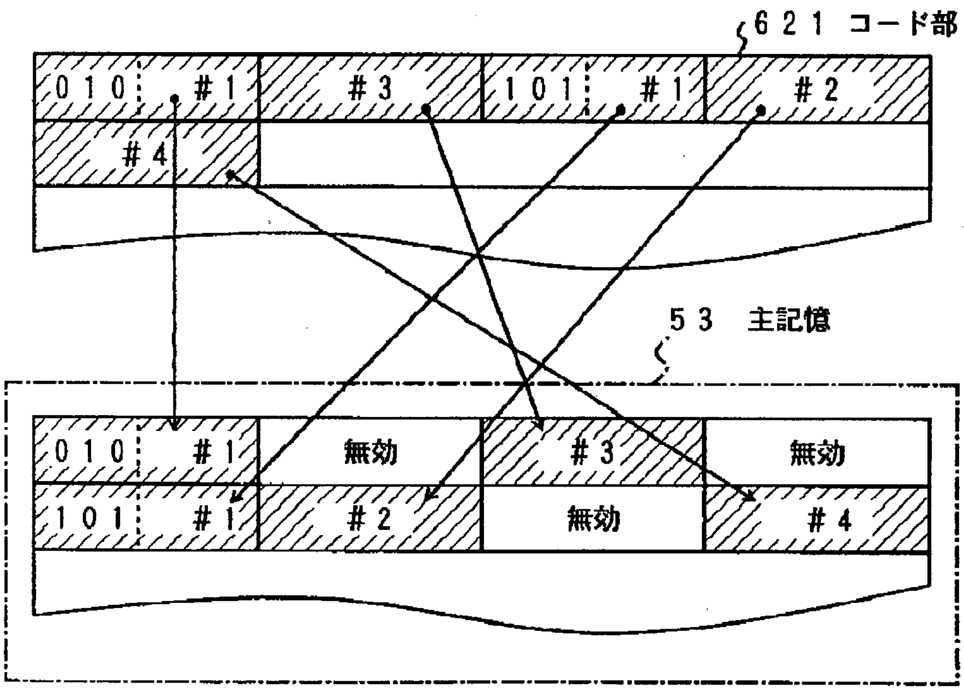
[Drawing 10]


1
ポインタを４バイト分進める
S51
i＝i＋1
S52
S53
i＞4
NO
2
YES
S54
ページエンド
？
NO
S55
YES
終　了
コード部６２１から次の命令（ＶＬＩＷの１番目の命令フィールドの内容）を取り出す
3

[Drawing 11]

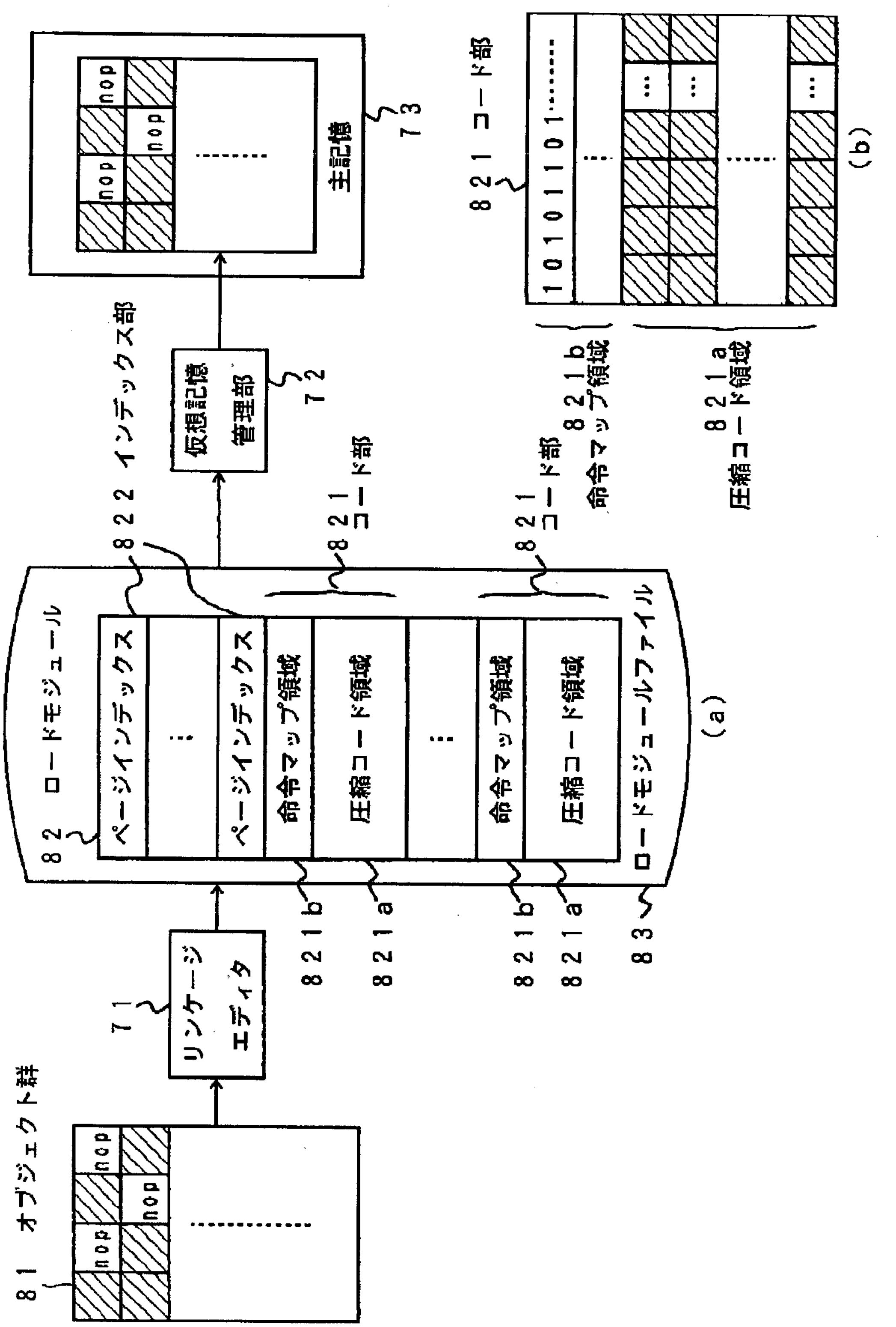


(a)



(b)

[Drawing 12]



[Translation done.]